UNITED STATES PATENT AND TRADEMARK OFFICE

A

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/896,526	06/28/2001	Haitham Akkary	42390P11201	8421

8791	7590	10/28/2005

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER
HUISMAN, DAVID J

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 10/28/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 August 2005*.

2a)☐ This action is **FINAL**.   2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7,9-11,13,15-22 and 24-35* is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7,9-11,13,15-22 and 24-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *25 March 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All   b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1.   Claims 1-7, 9-11, 13, 15-22, and 24-35 have been examined.

Papers Submitted

2.   It is hereby acknowledged that the following papers have been received and placed of

record in the file: Amendment as received on 8/11/2005 and RCE as received on 8/19/2005.

Claim Objections

3.   Claims 11, 20, and 29 are objected to because of the following informalities: In the last

paragraph of each claim, replace "and said single thread is not converted" to --said single thread

is not converted--. Appropriate correction is required.

Withdrawn Rejections

4.   Applicant, by way of amendment, has overcome the prior art rejections set forth in the

previous Office Action for each of the claims. Consequently, these rejections are hereby

withdrawn by the examiner and all arguments directed towards these rejections are moot.

However, upon further consideration, a new ground(s) of rejection is applied below.

Claim Rejections - 35 USC § 103

5.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in
> section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are
> such that the subject matter as a whole would have been obvious at the time the invention was made to a person

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6.   Claims 1-7, 9-10, and 29-35 are rejected under 35 U.S.C. 103(a) as being unpatentable

over Sundaramoorthy et al., "Slipstream Processors: Improving both Performance and Fault

Tolerance," ASPLOS, Nov. 2000 (as applied in the previous Office Action and herein referred to

as Sundaramoorthy) in view of Mukherjee, U.S. Patent No. 6,757,811 (as cited in a previous

Office Action) in view of Hennessy and Patterson, "Computer Architecture - A Quantitative

Approach, 2nd Edition," 1996 (as applied in the previous Office Action and herein referred to as

Hennessy).

7.   Referring to claim 1, Sundaramoorthy has taught an apparatus comprising:

a) a first processor and a second processor. See Fig.1 and note the R-stream and A-stream

processors.

b) a plurality of memory devices coupled to the first processor and the second processor. See

Fig.1 and note the I-cache and D-cache memories.

c) a first buffer coupled to the first processor and the second processor, the first buffer being a

register buffer. See Fig.1 and also see column 10, lines 17-21, and lines 35-38 and note that data

operands are passed from the A-stream processor to the R-stream processor via the delay buffer.

d) a second buffer coupled to the first processor and the second processor, the second buffer

being a trace buffer. See column 7, and note the paragraph beginning with bulleted paragraph

beginning with "Conventional Fetching...". In this paragraph a trace predictor/buffer is

disclosed which stores/buffers trace IDs.

e) a plurality of memory instruction buffers coupled to the first processor and the second

processor. Note from Fig.1 that separate reorder buffers are connected to each processor);

f) wherein the first processor and the second processor perform single threaded applications

using multithreading resources (col. 1, lines 53-54, col. 2, lines 18-20: teaches that a single

thread is instantiated twice such that two instances of the same thread exist and each instance is

executed by different processors).

g) the first processor executes a single threaded application ahead of the second processor

executing said single threaded application to avoid misprediction, and said single threaded

application is not converted to an explicit multiple thread application. See column 1, 2nd

paragraph, and note one is executed ahead of the other so that control outcomes may be passed to

the lagging thread. Also, see column 2, lines 37-43 and note that the R-stream receives accurate

predictions. Hence, branch mispredictions are avoided. Also, note that a single threaded

application is not converted to an explicit multiple-thread application. Instead, a single thread is

copied such that two instances of a single thread exist.

h) the single threaded application executed on the second processor avoids branch mispredictions

from information received from said first processor. See column 1, 2nd paragraph, and note one

is executed ahead of the other so that control outcomes may be passed to the lagging thread.

Also, see column 2, lines 37-43 and note that the R-stream receives accurate predictions. Hence,

branch mispredictions are avoided.

i) Sundaramoorthy has taught the need for a hardware monitor to detect ineffectual instructions

so that they may be bypassed in the leading A-stream (column 2, lines 23-32). This results in the

A-stream fetching, executing, and retiring fewer instructions than it would otherwise (column 2,

lines 34-35), thereby allowing the A-stream to stay ahead of the R-stream. In short,

Sundaramoorthy has taught that the A-stream and R-stream have different numbers of executed

instructions. Consequently, it follows that Sundaramoorthy has not taught that said single

threaded application contains the same number of instructions when executed on said first

processor and said second processor (as claimed by applicant). However, Mukherjee has taught

the concept of a single thread being executed twice in parallel as two threads, where the two

threads contain the same amount of instructions. See the abstract and Fig.3. A person of

ordinary skill in the art would've recognized that both Sundaramoorthy and Mukherjee have

taught redundant execution in order to speed up execution by passing information from one

stream to the other. The main difference is that Sundaramoorthy's leading stream runs ahead by

reducing the amount of instructions in the stream whereas Mukherjee's leading stream runs

ahead by merely starting execution earlier than the trailing stream (Mukherjee, Fig.3). By

modifying Sundaramoorthy to include the execution concept taught by Mukherjee, the hardware

monitor and speculative bypassing of instructions would be eliminated. This would in turn

eliminate bypassing errors that may occur (Sundaramoorthy, column 2, lines 45-50). As a result,

in order to eliminate the hardware monitor (and the problems that it may cause) from

Sundaramoorthy, it would have been obvious to one of ordinary skill in the art at the time of the

invention to modify Sundaramoorthy such that the exact same thread is executed twice, where

the leading thread is merely started before the trailing thread. It should further be noted that

while Mukherjee has taught SMT-style execution of two threads on a single processor (abstract),

the concept is easily applicable to a multiprocessor system. Sundaramoorthy even recognizes

this in column 2, lines 18-20, by saying that two redundant programs may execute on a

multiprocessor system or an SMT processing system, which is essentially like have multiple

processors on a single chip (virtual processors).

j) Sundaramoorthy has not taught that the first and second processors each have a scoreboard and a decoder.

However, Official Notice is taken that instruction decoders are well known and expected in the art. More specifically, after instructions are fetched by a processor, they must inherently be decoded so that the processor may determine what type of instruction has been fetched and consequently, what operation to perform. Clearly, if both processors of Sundaramoorthy are fetching instructions, then both sets must be decoded. As a result, it would have been obvious to have instruction decoders in each of the first and second processors so that instructions may be decoded. One would have been motivated to make such a modification to allow both processors to decode their own instructions.

In addition, Hennessy has taught that a scoreboard allows instructions to execute out of order. As is known in the art, out-of-order execution is advantageous because it allows instructions to execute as soon as their resources are ready, thereby reducing stalling and CPU idleness. See pages 241 and 242. As a result, in order to allow both processors to benefit from such execution and resulting advantages, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify each of the first and second processors of Sundaramoorthy to include scoreboards.

8.   Referring to claim 2, Sundaramoorthy in view of Mukherjee and further in view of Hennessy has taught an apparatus as described in claim 1. Sundaramoorthy has further taught that the memory devices comprise a plurality of cache devices (Fig. 1, I-Cache and D-Cache).

9.   Referring to claim 3, Sundaramoorthy in view of Mukherjee and further in view of Hennessy has taught an apparatus as described in claim 1. Sundaramoorthy has further taught

that the first processor is coupled to at least one of a plurality of zero level (L0) data cache

devices and at least one of a plurality of L0 instruction cache devices, and the second processor

is coupled to at least one of the plurality of L0 data cache devices and at least one of the plurality

of L0 instruction cache devices (fig. 1 shows that each processor is connected to a separate data

cache (D-Cache) and instruction (I-Cache) which can be considered as zero-level caches because

they are directly connected to the execute cores).

10.   Referring to claim 4, Sundaramoorthy in view of Mukherjee and further in view of

Hennessy has taught an apparatus as described in claim 3.   Sundaramoorthy has further taught

that each of the plurality of L0 data cache devices store exact copies of store instruction data.

Although this is not mentioned explicitly, it is deemed inherent to the design because as each

processor is executing the same thread (col. 1, lines 53-54, col. 2, lines 18-20) the data caches in

each processor must contain exact copies of data.   And, this data is store instruction data because

data that is stored to main memory is also stored in a data cache.

11.   Referring to claim 5, Sundaramoorthy in view of Mukherjee and further in view of

Hennessy has taught an apparatus as described in claim 1.   Sundaramoorthy has further taught

that the plurality of memory instruction buffers includes at least one store forwarding buffer (fig.

1, reorder buffer connected to A-stream processor) and at least one load-ordering buffer (fig. 1,

reorder buffer connected to R-stream processor).

12.   Referring to claim 6, Sundaramoorthy in view of Mukherjee and further in view of

Hennessy has taught an apparatus as described in claim 5.   Although Sundaramoorthy in view of

Mukherjee in view of Hennessy does not mention that the at least one store forwarding buffer

(fig. 1, reorder buffer (ROB) connected to A-stream processor) comprises a structure having a

plurality of entries, each of the plurality of entries having a tag portion, a validity portion, a data

portion, a store instruction identification (ID) portion, and a thread ID portion it is deemed

inherent to the design. A ROB is used to order instructions completing execution hence must

contain a plurality of entries. Also each entry must have a tag portion to index into the ROB, a

validity portion to indicate whether an entry can be written to or read from, a data portion for

storing the results of the instruction, a store instruction ID portion would be the instruction

opcode of an entry, and a thread ID for indicating which thread that instruction belongs to.

13.   Referring to claim 7, Sundaramoorthy in view of Mukherjee and further in view of

Hennessy has taught an apparatus as described in claim 6. Although Sundaramoorthy in view of

Mukherjee in view of Hennessy does not mention that the at least one load ordering buffer (fig.

1, reorder buffer connected to R-stream processor) comprises a structure having a plurality of

entries, each of the plurality of entries having a tag portion, an entry validity portion, a load

identification (ID) portion, and a load thread ID portion it is deemed inherent to the design. A

ROB is used to order instructions completing execution hence must contain a plurality of entries.

Also each entry must have a tag portion to index into the ROB, a validity portion to indicate

whether an entry can be written to or read from, a load instruction ID portion would be the

instruction opcode of an entry, and a thread ID for indicating which thread that instruction

belongs to.

14.   Referring to claim 9, Sundaramoorthy in view of Mukherjee and further in view of

Hennessy has taught an apparatus as described in claim 1. Furthermore, although

Sundaramoorthy has taught that the trace buffer (delay buffer) is a FIFO queue (col. 10, line 17),

they do not disclose that the trace buffer is a circular buffer having an array with head and tail

pointers, the head and tail pointers having a wrap-around bit. However, "Official Notice" is taken that it is well known and expected in the art to implement a FIFO queue as a circular buffer with head and tail pointers wherein head and tail pointers have a wrap-around bit. A circular buffer is useful to implement in hardware because only the head and tail pointers need to be incremented/decremented instead of actually physically shifting entries. A wrap around bit would also be needed to indicate whether the pointer has wrapped around the end of the queue. Therefore, it would been obvious to one of ordinary skill in the art at the time of the invention to have implemented the FIFO queue as a circular buffer with head and tail pointers, the head and tail pointers having a wrap around bit because it is known that a FIFO queue can be implemented as a circular buffer and it is easier to build in hardware.

15.   Referring to claim 10, Sundaramoorthy in view of Mukherjee and further in view of Hennessy has taught an apparatus as described in claim 1. Sundaramoorthy in view of Mukherjee in view of Hennessy has not explicitly taught that the register buffer comprising an integer register buffer and a predicate register buffer. However, Official Notice is taken that integer registers and predicate registers are well known and expected in the art. By implementing integer registers, the system will be able to load and store integer data and perform integer operations quickly. Furthermore, by implementing predicate registers, the system will be able to achieve conditional execution of instructions without conditional branch instructions. Consequently, to achieve such functionality, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sundaramoorthy in view of Mukherjee in view of Hennessy to include a integer register buffer and a predicate register buffer in the register buffer (delay buffer).

16.   Referring to claim 29, Sundaramoorthy has taught a system comprising:

a) a first processor (fig. 1, R-stream processor comprising of the execute core).

b) a second processor (fig. 1, A-stream processor comprising of the execute core).

c) a bus coupled to the first processor and the second processor (fig. 1, a bus is shown between

the first and second processors via the delay buffer);

d) a plurality of local memory devices coupled to the first processor and the second processor

(fig. 1, I-cache and D-cache memories);

e) a first buffer coupled to the first processor and the second processor, the first buffer being a

register buffer. See Fig.1 and also see column 10, lines 17-21, and lines 35-38 and note that data

operands are passed from the A-stream processor to the R-stream processor via the delay buffer.

d) a second buffer coupled to the first processor and the second processor, the second buffer

being a trace buffer. See column 7, and note the paragraph beginning with bulleted paragraph

beginning with "Conventional Fetching…". In this paragraph a trace predictor/buffer is

disclosed which stores/buffers trace IDs.

e) a plurality of memory instruction buffers coupled to the first processor and the second

processor. Note from Fig.1 that separate reorder buffers are connected to each processor);

f) wherein the first processor and the second processor perform single threaded applications

using multithreading resources (col. 1, lines 53-54, col. 2, lines 18-20: teaches that a single

thread is instantiated twice such that two instance of a single thread exist and each instance is

executed on a different processor).

g) the first processor executes a single threaded application ahead of the second processor

executing said single threaded application to avoid misprediction. See column 1, 2nd paragraph,

and note one is executed ahead of the other so that control outcomes may be passed to the

lagging thread. Also, see column 2, lines 37-43 and note that the R-stream receives accurate

predictions. Hence, branch mispredictions are avoided.

h) said single threaded application is not converted to an explicit multiple thread application.

Note that a single threaded application is not converted to multiple threads. Instead, a single

thread is copied such that two instances of a single thread exist.

i) the single threaded application executed on the second processor avoids branch mispredictions

from information received from said first processor. See column 1, 2nd paragraph, and note one

is executed ahead of the other so that control outcomes may be passed to the lagging thread.

Also, see column 2, lines 37-43 and note that the R-stream receives accurate predictions. Hence,

branch mispredictions are avoided.

j) Sundaramoorthy has taught the need for a hardware monitor to detect ineffectual instructions

so that they may be bypassed in the leading A-stream (column 2, lines 23-32). This results in the

A-stream fetching, executing, and retiring fewer instructions than it would otherwise (column 2,

lines 34-35), thereby allowing the A-stream to stay ahead of the R-stream. In short,

Sundaramoorthy has taught that the A-stream and R-stream have different numbers of executed

instructions. Consequently, it follows that Sundaramoorthy has not taught that said single

threaded application contains the same number of instructions when executed on said first

processor and said second processor (as claimed by applicant). However, Mukherjee has taught

the concept of a single thread being executed twice in parallel as two threads, where the two

threads contain the same amount of instructions. See the abstract and Fig.3. A person of

ordinary skill in the art would've recognized that both Sundaramoorthy and Mukherjee have

taught redundant execution in order to speed up execution by passing information from one

stream to the other. The main difference is that Sundaramoorthy's leading stream runs ahead by

reducing the amount of instructions in the stream whereas Mukherjee's leading stream runs

ahead by merely starting execution earlier than the trailing stream (Mukherjee, Fig.3). By

modifying Sundaramoorthy to include the execution concept taught by Mukherjee, the hardware

monitor and speculative bypassing of instructions would be eliminated. This would in turn

eliminate bypassing errors that may occur (Sundaramoorthy, column 2, lines 45-50). As a result,

in order to eliminate the hardware monitor (and the problems that it may cause) from

Sundaramoorthy, it would have been obvious to one of ordinary skill in the art at the time of the

invention to modify Sundaramoorthy such that the exact same thread is executed twice, where

the leading thread is merely started before the trailing thread. It should further be noted that

while Mukherjee has taught SMT-style execution of two threads on a single processor (abstract),

the concept is easily applicable to a multiprocessor system. Sundaramoorthy even recognizes

this in column 2, lines 18-20, by saying that two redundant programs may execute on a

multiprocessor system or an SMT processing system, which is essentially like have multiple

processors on a single chip (virtual processors).

k) Sundaramoorthy has not taught that the first and second processors each have a scoreboard

and a decoder.

However, Official Notice is taken that instruction decoders are well known and expected

in the art. More specifically, after instructions are fetched by a processor, they must inherently

be decoded so that the processor may determine what type of instruction has been fetched and

consequently, what operation to perform. Clearly, if both processors of Sundaramoorthy are

fetching instructions, then both sets must be decoded. As a result, it would have been obvious to

have instruction decoders in each of the first and second processors so that instructions may be

decoded. One would have been motivated to make such a modification to allow both processors

to decode their own instructions.

In addition, Hennessy has taught that a scoreboard allows instructions to execute out of

order. As is known in the art, out-of-order execution is advantageous because it allows

instructions to execute as soon as their resources are ready, thereby reducing stalling and CPU

idleness. See pages 241 and 242. As a result, in order to allow both processors to benefit from

such execution and resulting advantages, it would have been obvious to one of ordinary skill in

the art at the time of the invention to modify each of the first and second processors of

Sundaramoorthy to include scoreboards.

j) Sundaramoorthy also has not taught a main memory coupled to the bus. However, Official

Notice is taken that it is well known and expected in the art to have a main memory connected to

multiple processors via a common bus in a multi-processor environment. Since caches do not

store every instruction and data item, main memory must exist to store all of it. Therefore it

would have been obvious to one of ordinary skill in the art at the time of the invention to have

added a main memory coupled to the bus in the Sundaramoorthy reference.

17.   Referring to claim 30, Sundaramoorthy in view of Mukherjee in view of Hennessy has

taught a system as described in claim 29, wherein the memory devices comprise of a plurality of

cache devices (Fig. 1, I-Cache and D-Cache).

18.   Referring to claim 31, Sundaramoorthy in view of Mukherjee in view of Hennessy has

taught a system as described in claim 29, wherein the first processor is coupled to at least one of

a plurality of zero level (L0) data cache devices and at least one of a plurality of L0 instruction

cache devices, and the second processor is coupled to at least one of the plurality of L0 data

cache devices and at least one of the plurality of L0 instruction cache devices (fig. 1 shows that

each processor is connected to a separate data cache (D-Cache) and instruction (I-Cache) which

can be considered as zero-level caches because they are directly connected to the execute cores).

19.   Referring to claim 32, Sundaramoorthy in view of Mukherjee in view of Hennessy has

taught a system as described in claim 31, wherein each of the plurality of L0 data cache devices

store exact copies of store instruction data. Although this is not mentioned explicitly, it is

deemed inherent to the design because as each processor is executing the same thread (col. 1,

lines 53-54, col. 2, lines 18-20) the instruction and data caches in each processor must contain

exact copies of instructions and data. And, this data is store instruction data because data that is

stored to main memory is also stored in a data cache.

20.   Referring to claim 33, Sundaramoorthy in view of Mukherjee in view of Hennessy has

taught a system as described in claim 31. Sundaramoorthy in view of Mukherjee in view of

Hennessy has not taught that the first processor and the second processor each sharing a first

level (L1) cache device and a second level (L2) cache device. However, Official Notice is taken

that it is well known and expected in the art that processors in a multi-processor environment

share L1 and L2 cache devices. Such a scheme allows for the simplification of cache coherency

in that both processors would be able to access the same up-to-date cache as opposed to one of

the processors accessing out-of-date information in its own cache. Therefore, it would have been

obvious to one of ordinary skill in the art at the time of the invention to have the first and second

processors share L1 and L2 cache devices.

21.   Referring to claim 34, Sundaramoorthy in view of Mukherjee in view of Hennessy has taught a system as described in claim 29. Sundaramoorthy has further taught that the plurality of memory instruction buffers includes at least one store forwarding buffer (fig. 1,reorder buffer connected to A-stream processor) and at least one load-ordering buffer (fig. 1, reorder buffer connected to R-stream processor).

22.   Referring to claim 35, Sundaramoorthy in view of Mukherjee in view of Hennessy has taught a system as described in claim 34. Sundaramoorthy in view of Mukherjee in view of Hennessy has not taught that the at least one store forwarding buffer (fig. 1, reorder buffer (ROB) connected to A-stream processor) comprises a structure having a plurality of entries, each of the plurality of entries having a tag portion, a validity portion, a data portion, a store instruction identification (ID) portion, and a thread ID portion it is deemed inherent to the design. A ROB is used to order instructions completing execution hence must contain a plurality of entries. Also each entry must have a tag portion to index into the ROB, a validity portion to indicate whether an entry can be written to or read from, a data portion for storing the results of the instruction, a store instruction ID portion would be the instruction opcode of an entry, and a thread ID for indicating which thread that instruction belongs to.

23.   Claims 11, 13, and 15-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy in view of Mukherjee in view of Hennessy, as applied above, and further in view of Akkary, WO 99/31594 (as applied in the previous Office Action).

24.   Referring to claim 11, Sundaramoorthy has taught a method comprising:

a) executing a plurality of instructions in a single thread by a first processor (col. 1, lines 53-54,

col. 2, lines 18-23: The R-stream thread is executed by the R-stream processor in fig. 1).

b) executing said plurality of instructions in the single thread by a second processor (col. 1, lines

53-54, col. 2, lines 18-32: The A-stream thread, which is the same as the R-stream thread, is

executed by the A-stream processor) as directed by the first processor (col. 4, lines 21-38: IR-

detector and IR-predictor in fig. 1, which are part of the first processor i.e. R-stream processor,

direct the second processor (A-stream processor) to execute instructions from the A-stream), the

second processor executing said plurality of instructions ahead of the first processor (col. 2, lines

20-23: A-stream runs ahead of the R-stream and it is executed by the second processor to avoid

misprediction (See column 1, 2nd paragraph, and note one is executed ahead of the other so that

control outcomes may be passed to the lagging thread. Also, see column 2, lines 37-43 and note

that the R-stream receives accurate predictions. Hence, branch mispredictions are avoided.).

Note that even though less instructions may be executed by the first processor (due to removal),

all of the instructions executed by the first processor will be also be executed by the second

processor.

c) Sundaramoorthy has not taught tracking at least one register that is one of loaded from a

register file buffer, and written by said second processor, said tracking executed by said second

processor. However, Hennessy has taught the idea of a scoreboard which allows instructions to

execute out of order. As is known in the art, out-of-order execution is advantageous because it

allows instructions to execute as soon as their resources are ready, thereby reducing stalling and

CPU idleness. See pages 241 and 242. As a result, in order to allow the second processor to

benefit from such execution and resulting advantages, it would have been obvious to one of

ordinary skill in the art at the time of the invention to modify the second processor of

Sundaramoorthy to include a scoreboard. And, the inherent nature of a scoreboard is to track

registers written by the second processor. See Fig.4.4 on page 247, and note that the system

tracks when registers are ready so that execution may continue. For registers to be ready, it must

be tracked when the writing to those registers completes.

d) transmitting control flow information from the second processor to the first processor, the first

processor avoiding branch prediction by receiving the control flow information. See column 1,

2nd paragraph, column 2, lines 37-43, and column 11, line 5. Note that accurate control

information is sent to the R-stream so that predictions are not needed. The R-stream would

instead know which way to go from predictions in the A-stream.

e) transmitting results from the second processor to the first processor, the first processor

avoiding executing a portion of instructions (col. 10, lines 17-21, 30-33, 35-38: results (data-flow

information) are transmitted from the A-stream processor to the R-stream processor via the delay

buffer, and these values are used directly by the instructions hence avoiding the execution of the

portion of the instructions) by committing the results of the portion of instructions into a register

file from a first buffer, the first buffer being a trace buffer (Although this is not explicitly

mentioned, it is deemed inherent to the design because col. 4 line 15 discloses the presence of a

register file in the processor and as results are written to the register file so that they can be read

from by future instructions, the results of the instructions from the trace buffer (delay buffer)

must be written into the register file).

f) wherein the first processor and the second processor execute single threaded applications using

multithreading resources (col. 1, lines 53-54, col. 2, lines 18-20: teaches that a single thread is

instantiated twice such that two instances of the same thread exist and each instance is executed

by different processors), and said single threaded application is not converted to an explicit

multiple-thread application. Note that a single threaded application is not converted to multiple

threads. Instead, a single thread is copied such that two instances of a single thread exist.

g) the single threaded application executed on the second processor avoids branch mispredictions

using information received from said first processor. See column 1, 2nd paragraph, and note one

is executed ahead of the other so that control outcomes may be passed to the lagging thread.

Also, see column 2, lines 37-43 and note that the R-stream receives accurate predictions. Hence,

branch mispredictions are avoided.

h) Sundaramoorthy has taught the need for a hardware monitor to detect ineffectual instructions

so that they may be bypassed in the leading A-stream (column 2, lines 23-32). This results in the

A-stream fetching, executing, and retiring fewer instructions than it would otherwise (column 2,

lines 34-35), thereby allowing the A-stream to stay ahead of the R-stream. In short,

Sundaramoorthy has taught that the A-stream and R-stream have different numbers of executed

instructions. Consequently, it follows that Sundaramoorthy has not taught that said single

threaded application contains the same number of instructions when executed on said first

processor and said second processor (as claimed by applicant). However, Mukherjee has taught

the concept of a single thread being executed twice in parallel as two threads, where the two

threads contain the same amount of instructions. See the abstract and Fig.3. A person of

ordinary skill in the art would've recognized that both Sundaramoorthy and Mukherjee have

taught redundant execution in order to speed up execution by passing information from one

stream to the other. The main difference is that Sundaramoorthy's leading stream runs ahead by

reducing the amount of instructions in the stream whereas Mukherjee's leading stream runs

ahead by merely starting execution earlier than the trailing stream (Mukherjee, Fig.3). By

modifying Sundaramoorthy to include the execution concept taught by Mukherjee, the hardware

monitor and speculative bypassing of instructions would be eliminated. This would in turn

eliminate bypassing errors that may occur (Sundaramoorthy, column 2, lines 45-50). As a result,

in order to eliminate the hardware monitor (and the problems that it may cause) from

Sundaramoorthy, it would have been obvious to one of ordinary skill in the art at the time of the

invention to modify Sundaramoorthy such that the exact same thread is executed twice, where

the leading thread is merely started before the trailing thread. It should further be noted that

while Mukherjee has taught SMT-style execution of two threads on a single processor (abstract),

the concept is easily applicable to a multiprocessor system. Sundaramoorthy even recognizes

this in column 2, lines 18-20, by saying that two redundant programs may execute on a

multiprocessor system or an SMT processing system, which is essentially like have multiple

processors on a single chip (virtual processors).

i) Sundaramoorthy in view of Mukherjee in view of Hennessy has not taught clearing a store

validity bit and setting a mispredicted bit in a load entry in the first buffer if a replayed store

instruction has a matching store identification (ID) portion in a second buffer, the second buffer

being a load buffer. However, Official Notice is taken that it is well known and expected in the

art to use load and store buffers for the proper handling of memory operations. Akkary discloses

a system for ordering loads and stores in a multithreaded processor using load and store buffers

(fig. 2, 182,184). He discloses clearing a store validity bit (SB Hit field) in the load buffer if data

came from memory (pg. 37, para. 3, line 4; pg. 38, line 1). Also when a store instruction is

executed (which includes replayed stores), its address is compared with the store ID portion

(addresses) of load instructions (pg. 36, para. 3). On a match, a replay event is signaled to the

load entry in the trace buffer to replay the load instruction and all its dependant instructions

because it was mispredicted (pg. 38, para. 2). Furthermore, Official Notice is taken that is well

known and expected in the art to set a status bit to indicate a misprediction. Clearly, in order to

detect a misprediction, some bit must change somewhere in the system. As shown in In re

Larson, 144 USPQ 347 (CCPA 1965), to make integral is generally not given patentable weight

or would have been an obvious improvement. That is, it does not matter where this

misprediction bit is located within the system, as long as it exists. One of ordinary skill in the art

would have recognized that one could use the load and store buffer arrangement of Akkary in the

Sundaramoorthy reference in order handle loads and stores in the multithreaded environment.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the

invention to have modified the Sundaramoorthy reference by clearing a store validity bit and

setting a mispredicted bit in a load entry in the trace buffer (delay buffer) if a replayed store

instruction has a matching store ID portion.

25.   Referring to claim 13, Sundaramoorthy in view of Mukherjee in view of Hennessy and

further in view of Akkary has taught a method as described in claim 11. Sundaramoorthy has

further taught duplicating memory information in separate memory devices for independent

access by the first processor and the second processor. Although this is not mentioned explicitly,

it is deemed inherent to the design because as each processor is executing the same thread (col.

1, lines 53-54, col. 2, lines 18-20) the instruction and data caches in each processor (fig. 1) must

contain exact copies of instructions and data.

26.   Referring to claim 15, Sundaramoorthy in view of Mukherjee in view of Hennessy and

further in view of Akkary has taught a method as described in claim 11. Sundaramoorthy in

view of Mukherjee in view of Hennessy has not taught setting a store validity bit if a store

instruction that is not replayed matches a store identification (ID) portion. However, Official

Notice is taken that it is well known and expected in the art to use load and store buffers for the

proper handling of memory operations. Akkary discloses a system for ordering loads and stores

in a multithreaded processor using load and store buffers (fig. 2, 182,184). He discloses setting a

store validity bit (SB Hit field) in the load buffer if data came from store buffer (pg. 37, para. 3,

line 4; pg. 38, lines 1-2). In order for data to come from the store buffer, a store instruction

address (including store instructions that are not replayed) must match a store ID portion

(address) of the load entry. One of ordinary skill in the art would have recognized that one could

use the load and store buffer arrangement of Akkary in the Sundaramoorthy reference in order

handle loads and stores in the multithreaded environment. Therefore it would have been obvious

to one of ordinary skill in the art at the time of the invention to have modified the

Sundaramoorthy reference by setting a store validity bit if a store instruction that is not replayed

matches the store ID portion.

27.   Referring to claim 16, Sundaramoorthy in view of Mukherjee in view of Hennessy and

further in view of Akkary has taught a method as described in claim 11. Furthermore, although

Sundaramoorthy has taught flushing the pipeline (reorder buffer) of the R-stream on a

misprediction, Sundaramoorthy has not taught flushing a pipeline, setting a mispredicted bit in a

load entry in the trace buffer and restarting a load instruction if one of the load is not replayed

and does not match a tag portion in a load buffer, and the load instruction matches the tag portion

in the load buffer while a store valid bit is not set. However, Official Notice is taken that it is

well known and expected in the art to use load and store buffers for the proper handling of

memory operations. Akkary discloses a system for ordering loads and stores in a multithreaded

processor using load and store buffers (fig. 2, 182,184). In particular, when a store valid bit is not

set (SB hit = 0, pg. 38, para. 2) and when a store instruction compared with the addresses of load

instructions (pg. 36, para. 3) is a match, a replay event is signaled to the load entry in the trace

buffer to replay the load instruction and all its dependant instructions because it was

mispredicted (pg. 38, para. 2). Furthermore, Official Notice is taken that is well known and

expected in the art to set a status bit to indicate a misprediction. Clearly, in order to detect a

misprediction, some bit must change somewhere in the system. As shown in In re Larson, 144

USPQ 347 (CCPA 1965), to make integral is generally not given patentable weight or would

have been an obvious improvement. That is, it does not matter where this misprediction bit is

located within the system, as long as it exists. One of ordinary skill in the art would have

recognized that one could use the load and store buffer arrangement of Akkary in the

Sundaramoorthy reference in order handle loads and stores in the multithreaded environment and

flush the pipeline on reading the mispredicted bit. Therefore it would have been obvious to one

ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy

reference by flushing a pipeline, setting a mispredicted bit in a load entry in the trace buffer and

restarting a load instruction if one of the load is not replayed and does not match a tag portion in

a load buffer, and the load instruction matches the tag portion in the load buffer while a store

valid bit is not set.

28.   Referring to claim 17, Sundaramoorthy in view of Mukherjee in view of Hennessy and

further in view of Akkary has taught a method as described in claim 11. Sundaramoorthy has

further taught executing a replay mode at a first instruction of a speculative thread (col. 1, lines

53-54, col. 2, lines 18-20: This feature is deemed inherent to the reference because when the A-

stream is initially started, i.e., at the first instruction, there will be two redundant threads being

executed which means the thread is being replayed from that point. This can be called a replay

mode).

29.   Referring to claim 18, Sundaramoorthy in view of Mukherjee in view of Hennessy and

further in view of Akkary has taught a method as described in claim 11. Sundaramoorthy has

further taught:

a) issuing all instructions up to a next replayed instruction including dependent instructions (This

feature is deemed inherent to the design because in order to execute the thread all instructions are

issued in either one of the R-stream and A-stream processors).

b) issuing instructions that are not replayed as no-operation (NOPs) instructions (This feature is

also deemed inherent to the design because if an instruction that is not replayed does not occupy

a slot in the execution pipeline it will lead to improper functioning of the processor. Hence as the

instruction that is not replayed is not to be executed, a NOP must be issued in its place).

c) issuing all load instructions and store instructions to memory (This limitation is also deemed

inherent to the design because if all loads and stores are not issued to memory, the state of the

thread would be incorrect leading to the malfunctioning of the system).

d) committing non-replayed instructions from the trace buffer to the register file (Although this is

not explicitly mentioned, it is deemed inherent to the design because col. 4 line 15 discloses the

presence of a register file in the processor and as results are written to the register file so that

they can be read from by future instructions, the results of the instructions from the trace buffer

(delay buffer) that are not going to be replayed must be written into the register file).

e) Sundaramoorthy in view of Mukherjee in view of Hennessy has not taught supplying names

from the trace buffer to preclude register renaming. However, Hennessy has taught that register

renaming is used to reduce name dependencies allowing instructions involved in name

dependencies to execute simultaneously or be reordered (pg. 232, para. 5). As these

dependencies are resolved, more instruction level parallelism can be extracted and performance

can be improved. One of ordinary skill in the art would have recognized to use register renaming

in the Sundaramoorthy reference because it too would improve performance. As the trace buffer

(delay buffer) would also supply the names, it would be logical not to do the renaming again in

the R-stream processor. Therefore, it would have been obvious to one of ordinary skill in the art

at the time of the invention to have modified the Sundaramoorthy reference by adding register

renaming capabilities and supply names from the trace buffer to preclude register renaming. One

would have been motivated to do so because it would improve performance which is one of the

objectives of the Sundaramoorthy reference (col. 1, lines 29-36).

30.   Referring to claim 19, Sundaramoorthy in view of Mukherjee in view of Hennessy and

further in view of Akkary has taught a method as described in claim 11. Sundaramoorthy has

further taught clearing a valid bit in an entry in a load buffer (fig. 1, the reorder buffer connected

to the R-stream processor) if the load entry is retired (Although not explicitly mentioned, it is

deemed inherent to the design because a load entry, on being retired, has to be marked invalid to

ensure that other new instructions can occupy that entry safely).

31.   Claims 20-22 and 24-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over

Sundaramoorthy in view of Mukherjee in view of Hennessy in view of Akkary, as applied above,

and further in view of Tanenbaum, "Structured Computer Organization," Prentice-Hall, 1984,

pp. 10-12 (as applied in the previous Office Action and herein referred to as Tanenbaum).

32.   Referring to claim 20, Sundaramoorthy has taught:

a) executing a single thread from a first processor (col. 1, lines 53-54, col. 2, lines 18-23: The R-

stream thread is executed by the R-stream processor in fig. 1).

b) executing said single thread from a second processor (col. 1, lines 53-54, col. 2, lines 18-32:

The A-stream thread, which is the same as the R-stream thread, is executed by the A-stream

processor) as directed by the first processor (col. 4, lines 21-38: IR-detector and IR-predictor in

fig. 1, considered part of the first processor i.e. R-stream processor, direct the second processor

(A-stream processor) to execute instructions from the A-stream), the second processor executing

instructions ahead of the first processor to avoid misprediction (See column 1, 2nd paragraph,

and note one is executed ahead of the other so that control outcomes may be passed to the

lagging thread. Also, see column 2, lines 37-43 and note that the R-stream receives accurate

predictions. Hence, branch mispredictions are avoided.).

c) Sundaramoorthy has not taught tracking at least one register that is one of loaded from a first

buffer, and written by said second processor, said tracking executed by said second processor,

the first buffer being a register file buffer. However, Hennessy has taught the idea of a

scoreboard which allows instructions to execute out of order. As is known in the art, out-of-

order execution is advantageous because it allows instructions to execute as soon as their

resources are ready, thereby reducing stalling and CPU idleness. See pages 241 and 242. As a

result, in order to allow the second processor to benefit from such execution and resulting

advantages, it would have been obvious to one of ordinary skill in the art at the time of the

invention to modify the second processor of Sundaramoorthy to include a scoreboard. And, the

inherent nature of a scoreboard is to track registers written by the second processor. See Fig.4.4

on page 247, and note that the system tracks when registers are ready so that execution may

continue. For registers to be ready, it must be tracked when the writing to those registers

completes.

d) wherein the first processor and the second processor execute single threaded applications

using multithreading resources (col. 1, lines 53-54, col. 2, lines 18-20: teaches that a single

thread is instantiated twice such that two instances of the same thread exist and each instance is

executed by different processors), and said single threaded application is not converted to an

explicit multiple-thread application. Note that a single threaded application is not converted to

multiple threads. Instead, a single thread is copied such that two instances of a single thread

exist.

e) the single threaded application executed on the second processor avoids branch mispredictions

using information received from said first processor. See column 1, 2nd paragraph, and note one

is executed ahead of the other so that control outcomes may be passed to the lagging thread.

Also, see column 2, lines 37-43 and note that the R-stream receives accurate predictions. Hence,

branch mispredictions are avoided.

f) Sundaramoorthy has taught the need for a hardware monitor to detect ineffectual instructions

so that they may be bypassed in the leading A-stream (column 2, lines 23-32). This results in the

A-stream fetching, executing, and retiring fewer instructions than it would otherwise (column 2,

lines 34-35), thereby allowing the A-stream to stay ahead of the R-stream. In short,

Sundaramoorthy has taught that the A-stream and R-stream have different numbers of executed

instructions. Consequently, it follows that Sundaramoorthy has not taught that said single

threaded application contains the same number of instructions when executed on said first

processor and said second processor (as claimed by applicant). However, Mukherjee has taught

the concept of a single thread being executed twice in parallel as two threads, where the two

threads contain the same amount of instructions. See the abstract and Fig.3. A person of

ordinary skill in the art would've recognized that both Sundaramoorthy and Mukherjee have

taught redundant execution in order to speed up execution by passing information from one

stream to the other. The main difference is that Sundaramoorthy's leading stream runs ahead by

reducing the amount of instructions in the stream whereas Mukherjee's leading stream runs

ahead by merely starting execution earlier than the trailing stream (Mukherjee, Fig.3). By

modifying Sundaramoorthy to include the execution concept taught by Mukherjee, the hardware

monitor and speculative bypassing of instructions would be eliminated. This would in turn

eliminate bypassing errors that may occur (Sundaramoorthy, column 2, lines 45-50). As a result,

in order to eliminate the hardware monitor (and the problems that it may cause) from

Sundaramoorthy, it would have been obvious to one of ordinary skill in the art at the time of the

invention to modify Sundaramoorthy such that the exact same thread is executed twice, where

the leading thread is merely started before the trailing thread. It should further be noted that

while Mukherjee has taught SMT-style execution of two threads on a single processor (abstract),

the concept is easily applicable to a multiprocessor system. Sundaramoorthy even recognizes

this in column 2, lines 18-20, by saying that two redundant programs may execute on a

multiprocessor system or an SMT processing system, which is essentially like have multiple

processors on a single chip (virtual processors).

g) Sundaramoorthy in view of Mukherjee in view of Hennessy has not taught clearing a store

validity bit and setting a mispredicted bit in a load entry in a second buffer if a replayed store

instruction has a matching store identification (ID) portion, the second buffer being a trace

buffer. However, Official Notice is taken that it is well known and expected in the art to use

load and store buffers for the proper handling of memory operations. Akkary discloses a system

for ordering loads and stores in a multithreaded processor using load and store buffers (fig. 2,

182,184). He discloses clearing a store validity bit (SB Hit field) in the load buffer if data came

from memory (pg. 37, para. 3, line 4; pg. 38, line 1). Also when a store instruction is executed

(which includes replayed stores), its address is compared with the store ID portion (addresses) of

load instructions (pg. 36, para. 3). On a match, a replay event is signaled to the load entry in the

trace buffer to replay the load instruction and all its dependant instructions because it was

mispredicted (pg. 38, para. 2). Furthermore, Official Notice is taken that is well known and

expected in the art to set a status bit to indicate a misprediction. Clearly, in order to detect a

misprediction, some bit must change somewhere in the system. As shown in In re Larson, 144

USPQ 347 (CCPA 1965), to make integral is generally not given patentable weight or would

have been an obvious improvement. That is, it does not matter where this misprediction bit is

located within the system, as long as it exists. One of ordinary skill in the art would have

recognized that one could use the load and store buffer arrangement of Akkary in the

Sundaramoorthy reference in order handle loads and stores in the multithreaded environment.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the

invention to have modified the Sundaramoorthy reference by clearing a store validity bit and

setting a mispredicted bit in a load entry in the trace buffer (delay buffer) if a replayed store

instruction has a matching store ID portion.

h) Sundaramoorthy in view of Mukherjee in view of Hennessy has not taught an apparatus

comprising a machine-readable medium containing instructions which, when executed by a

machine to perform the aforementioned operations. However, Tanenbaum has taught that any

instruction executed by hardware can also be simulated in software (pg 11, para. 4, lines 1-2).

He also teaches that hardware is generally immutable (first para. after sec. 1.4 header) while

software allows for more rapid change (pg. 11, para. 4, lines 2-4). One of ordinary skill in the art

at the time of the invention would have been motivated to convert the Sundaramoorthy reference

to software i.e. instructions on a machine readable medium because Tanenbaum teaches that

hardware is generally immutable (first para. after sec. 1.4 header) while software allows for more

rapid change (pg. 11, para. 4, lines 2-4). Therefore, to allow for ease of correction of mistakes,

and/or an ease of addition of new functionality, it would have been obvious to one of ordinary

skill in the art to have implemented the method of Sundaramoorthy by an apparatus comprising

instructions recorded on a machine readable medium.

33.   Referring to claim 21, Sundaramoorthy in view of Mukherjee in view of Hennessy in

view of Akkary and further in view of Tanenbaum has taught an apparatus as described in claim

20. Sundaramoorthy has further taught transmitting control flow information from the second

processor to the first processor, the first processor avoiding branch prediction by receiving the

control flow information (col. 10, lines 17-21, 30-35, 43-46).

34.   Referring to claim 22, Sundaramoorthy in view of Mukherjee in view of Hennessy in

view of Akkary and further in view of Tanenbaum has taught an apparatus as described in claim

21. Sundaramoorthy has further taught duplicating memory information in separate memory

devices for independent access by the first processor and the second processor (Although this is

not mentioned explicitly, it is deemed inherent to the design because as each processor is

executing the same thread (col. 1, lines 53-54, col. 2, lines 18-20) the instruction and data caches

in each processor (fig. 1) must contain exact copies of instructions and data).

35.   Referring to claims 24-25, Sundaramoorthy in view of Mukherjee in view of Hennessy in

view of Akkary and further in view of Tanenbaum has taught an apparatus as described in claim

21. Furthermore, claims 24-25 are rejected for the same reasons set forth in the rejections of

claims 15-16, respectively.

36.   Referring to claim 26, Sundaramoorthy in view of Mukherjee in view of Hennessy in

view of Akkary and further in view of Tanenbaum has taught an apparatus as described in claim

21. Sundaramoorthy has further taught:

a) executing a replay mode at a first instruction of a speculative thread (col. 1, lines 53-54, col. 2,

lines 18-20: This feature is deemed inherent to the reference because when the A-stream is

initially started i.e. at the first instruction, there will be two redundant threads being executed

which means the thread is being replayed from that point. This can be called a replay mode).

b) terminating the replay mode and the execution of the speculative thread if a partition in the

second buffer is approaching an empty state (this limitation is also deemed inherent to the

reference because when the partition in the trace buffer (delay buffer col. 10 lines 15+) is

approaching an empty state that means the A-stream has stopped producing results and finished

executing. Therefore now the replay mode and the A-stream are terminated).

37.   Referring to claim 27, Sundaramoorthy in view of Mukherjee in view of Hennessy in

view of Akkary and further in view of Tanenbaum has taught a method as described in claim 21.

Sundaramoorthy has further taught:

a) issuing all instructions up to a next replayed instruction including dependent instructions (This

feature is deemed inherent to the design because in order to execute the thread all instructions are

issued in either one of the R-stream and A-stream processors).

b) issuing instructions that are not replayed as no-operation (NOPs) instructions (This feature is

also deemed inherent to the design because if an instruction that is not replayed does not occupy

a slot in the execution pipeline it will lead to improper functioning of the processor. Hence as the

instruction that is not replayed is not to be executed, a NOP must be issued in its place).

c) issuing all load instructions and store instructions to memory (This limitation is also deemed

inherent to the design because if all loads and stores are not issued to memory, the state of the

thread would be incorrect leading to the malfunctioning of the system).

d) committing non-replayed instructions from the second buffer to a register file (Although this

is not explicitly mentioned, it is deemed inherent to the design because col. 4 line 15 discloses

the presence of a register file in the processor and as results are written to the register file so that

they can be read from by future instructions, the results of the instructions from the trace buffer

(delay buffer) that are not going to be replayed must be written into the register file).

e) Sundaramoorthy in view of Mukherjee in view of Hennessy has not taught supplying names

from the second buffer to preclude register renaming. However, Hennessy has taught that

register renaming is used to reduce name dependencies allowing instructions involved in name

dependencies to execute simultaneously or be reordered (pg. 232, para. 5). As these

dependencies are resolved, more instruction level parallelism can be extracted and performance

can be improved. One of ordinary skill in the art would have recognized to use register renaming

in the Sundaramoorthy reference because it too would improve performance. As the trace buffer

(delay buffer) would also supply the names, it would be logical not to do the renaming again in

the R-stream processor. Therefore, it would have been obvious to one of ordinary skill in the art

at the time of the invention to have modified the Sundaramoorthy reference by adding register

renaming capabilities and supply names from the trace buffer to preclude register renaming. One

would have been motivated to do so because it would improve performance which is one of the

objectives of the Sundaramoorthy reference (col. 1, lines 29-36).

38.   Referring to claim 28, Sundaramoorthy in view of Mukherjee in view of Hennessy in

view of Akkary and further in view of Tanenbaum has taught an apparatus as described in claim

21. Sundaramoorthy has further taught clearing a valid bit in an entry in a load buffer (fig. 1, the

reorder buffer connected to the R-stream processor) if the load entry is retired (Although not

explicitly mentioned, it is deemed inherent to the design because a load entry, on being retired,

has to be marked invalid to ensure that other new instructions can occupy that entry safely).
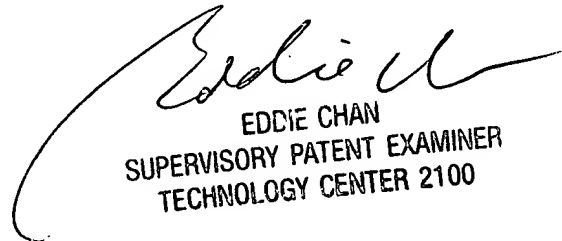

Conclusion

Any inquiry concerning this communication or earlier communications from the

examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168.

The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the

organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent

Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR

system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
October 19, 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100